Climbing and Walking Robots: Proceedings of the 7th International Conference CLAWAR 2004. Springer, 2005. 869-878

**DOI:** http://dx.doi.org/10.1007/3-540-29461-9_85

# Locomotion of a Modular Worm-like Robot using a FPGA-based embedded MicroBlaze Soft-processor

González-Gómez J., Aguayo E., and Boemo E.

Escuela Politécnica Superior, Universidad Autónoma de Madrid, Spain
{Juan.Gonzalez, Estanislao.Aguayo, Eduardo.Boemo}@uam.es

**Abstract.** Modular reconfigurable robots offer the promise of more versatility, robustness, and low cost. They are composed of simple and small modules, capable of attach and detach one to each other. In this paper, a modular worm-like robot composed of a chain of 8 similar modules is presented. A travelling wave, that moves from the tail to the head, propels the robot forward. The positions of the articulations are calculated using the following parameters: waveform, amplitude, and wavelength. Instead of a conventional architecture, a FPGA-based soft-processor core is utilized. It includes a set of custom peripheral cores, written in VHDL. FPGAs make modular robots more versatile, adding some new featureas to the design of robots like reconfigurable control, hardware reuse, lower cost, fault-recovering, and software/hardware co-design.

## 1 Introduction

Modular self-reconfigurable robots offer the promise of more versatility, robustness and low cost[1]. They are composed of modules, capable of attach and detach one to each other, changing the shape of the robot. This scheme allows them to perform unusual actions like to traverse through any kind of terrain as well as climbing over obstacles or crawling inside tubes. Utilities outside the research world has not been seen yet, but they are planned to be used in space applications[3] and urban search and rescue[2].

A modular robot with N different types of modules is called N-modular. Heterogeneity is tend to be reduced, decreasing the ratio between N and the total number of modules. In the last years, the number of robot following this approach has growth substantially[5][6][7][8].

One of the most advanced systems is Polybot[1][4], a 2-modular reconfigurable robot. Different reconfigurations and gaits has been probed; for example, from a loop, that uses a rolling gait, to a snake, with a sinusoidal gait, and finally to an spider. Currently, the third generation of modules (G3) is being developed[9]. Each module has its own embebed PowerPC 555 processor with a traditional processor architecture.

An addtional step on modurality is the use of FPGA technology instead of a conventional microprocessor chip. It gives the designer the possibility of implementing new architectures, faster control algorithms, or dinamically modify the hardware to adapt it to a new situation. In summary, Modular Reconfigurable Robot controlled by a FPGA not only are able to change their shapes, but also their hardware, so that, complete versatility can be achieved.

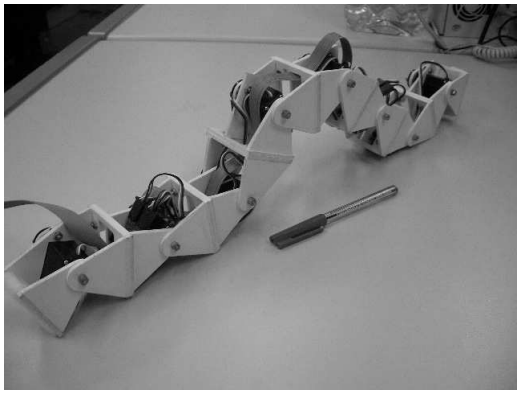

**Fig. 1.** The worm-like modular robot Cube. It is composed of 8 similar linked modules, connected in phase.

In this paper, a modular worm-like robot (figure 1), named Cube is presented. This is the simplest kind of modular robot, composed of 8 equal linked modules (1-modular robot). The locomotion is achieved by the propagation of waves that travel through the robot, from tail to the head. The entirely locomotion controller is embedded into an FPGA.

In this first prototype, the problem of worms locomotion and FPGA-based control has been solved. The control system is centralized; an unique custom FPGA processor can control the 8 modules.

The MicroBlaze soft-processor[10] has been selected as core processor. Additional hardware units has been implemented as VHDL modules. MicroBlaze is a powerful 32-bit processor, that offers new capabilities not available on conventional processors, like the addition of custom peripheral, duplicated modules to increase reliability, or the use of dynamic reconfiguration to adapt the control to a new enviroment. This soft-processor can also run operating systems like uC/OS-II[11], a real time OS, or uCLinux[12].

The organization of the paper is as follows. Firstly, the mechanics and the modules is presented. Secondly, the robot locomotion, algorithms, and the locomotion controller is addressed. Finally, the implementation on FPGA is explained and the results are presented.

## 2 Mechanical description

The current version of the prototype is a chain of 8 similar linked modules called Y1. In figure 2, a CAD rendering is showed. They have just one degree of freedom, actuated by a Futaba 3003 RC servo. The design is based on generation G1 Polybot modules[4]. In this first version, no sensors are included. Our main interest was focused on the study of locomotion, and its implementation on FPGA.

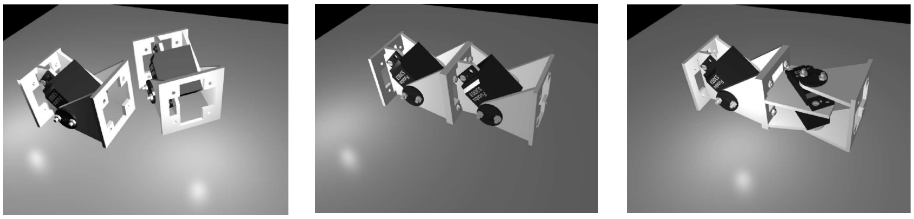

**Fig. 2.** CAD rendering of two Y1 modules. On the left, Two isolate modules In the middle, connection in phase. On the right, they are connected out of phase.

Y1 modules are simple and cheap: it is very easy to build prototypes of worm-like robots with them. These modules can be connected in two different ways, as shown in figure 2. One way is the connection in phase, in which two adjacent modules have the same orientation. Robots constructed using this link have all the articulations in the same plane, perpendicular to the ground (figure 3). Cube comprises 8 Y1 modules, connected in phase, so that it can only move along a line, forward or backwards.

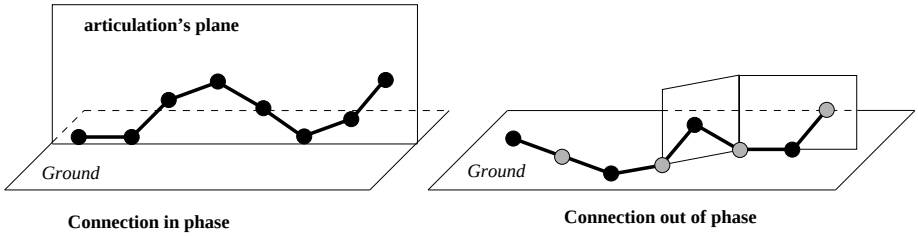


**Fig. 3.** Two schemes of worm-like robots. On the left, all the articulations are connected in phase so that they all are on the same plane. On the right, the connection is out of phase. With this configuration, the modules can be on different planes.

The other way of connecting the modules is out of phase. Two adjacent modules are rotated 90 degrees one to each other, obtaining two degrees of freedom. One articulation moves on the ground plane (yaw) and the other does perpendicularly (pitch). The right image of figure 3 shows a worm with this kind of links. Black circles represent articulations that moves on the ground plane and grey circles represents articulations that moves perpendicular. This kind of robot can turn and move on different directions, not just in straight line.

The dimensions of each module, in its initial position (0 degrees angle), are 52 x 52 x 72mm, and the weight is 50gr. They are made out of PVC. The rotations range is between -90 and 90 degrees. The robot is 576mm in length and 400gr in weight. The electronic and power supply are located off-board.

The consumption depends on the gaits, but typically it is 200mA per servo, giving a total of 1.6A. All the locomotion experiment at this first stage are realized using an off-board power supply.

## 3 Locomotion

Locomotion is achieved by the propagation of waves that traverse the worm, from the tail to the head. For programming simplicity, gait control tables are used[1], described in more detail in section 3.1. The locomotion controller (section 3.3) generates these tables automatically. The position controller reads them, producing the PWM signals to actuate the servos, and thus propelling the robot.

### 3.1 Gait control tables

Each articulation is characterized by the angle between the two segments it links. The shape of the worm, at a given instant $t$, is determined by the angular position vector $\overrightarrow{\varphi(t)} = (\varphi_1, \varphi_2, ..., \varphi_n)$. Figure 4 shows a six articulations worm-like robot and the angular position vector at a given time.

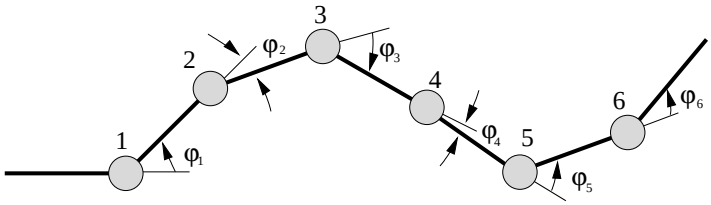


**Fig. 4.** Angular position vector for a worm-like robot composed of 6 articulations: $\overrightarrow{v} = (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6)$

For every instant, an angular position vector there exist, determining the shape of the worm: $\overrightarrow{\varphi(t_0)}, \overrightarrow{\varphi(t_1)}, ..., \overrightarrow{\varphi(t_m)}$. The control table is a matrix, which rows contains the angular position vectors for every instant. In order to generate the movement, the controller has to read the table, row by row, positioning the servos.

In robots like Polybot, this tables are pre-calculated and downloaded into the modules. Each table represents one gait. It is not possible to calculate or store all the possible tables for all the different gaits. Those tables are generated automatically in *Cube*.

### 3.2 Automatic generation of gait control tables

Control tables are generated using a wave propagation model. The algorithm is as follows (figure 5). Having a waveform in its initial state, $f(x, t_0)$ (in the figure, sinusoidal

waves are drawn, but other waveforms could be used) and a worm with all its articulations over the *x* axis (figure 5-1). Let $(x_i, y_i)$ be the coordinates of the articulation *i*, at some instant $t$. The angular position vector for the initial time, $\overrightarrow{\varphi(t_o)}$, is calculated fitting the articulations to the wave, so that $y_i = f(x_i, t_0)$ for all i. The distance L between articulations is maintained. It could be said that "the worm fits the wave" (figure 5-2). Next, the wave is shifted (instant $t_1$. Figure 5-3) and the worm fits the wave again, obtaining $\overrightarrow{\varphi(t_1)}$ (figure 5-4). Points 3 and 4 are repeated until the wave reach its initial phase. After *m* instant of time, all the vector that comprises the table are generated.

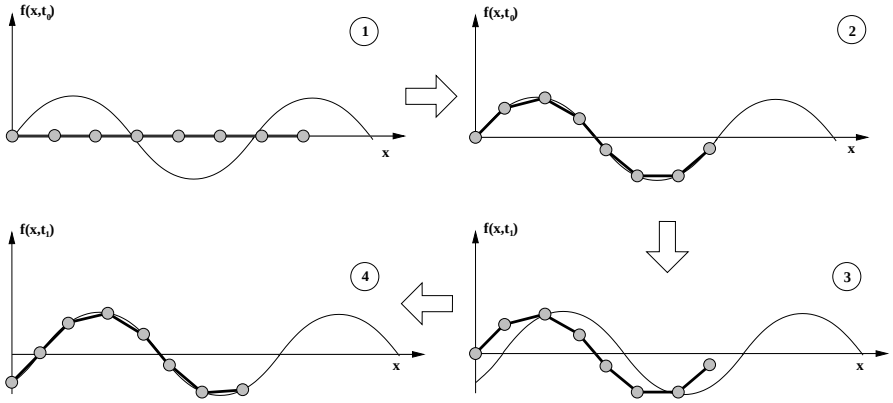


**Fig. 5.** The algorithm used to generate the control tables

By means of this algorithm, control tables are obtained, regardless of the waveform used, $f(x,t)$. In the locomotion test, sinusoidal and semi-sinusoidal waves (just the positive part of the sinusoidal wave) have been used.

### 3.3 Locomotion controller

The locomotion controller generates the PWM signals for positioning the servos from the wave parameters: waveform, amplitude, and wavelength. Higher level systems could move the robot just specifying this parameters. Furthermore, at this stage, the movement of the robot is independent of the number of articulations. The planificator algorithm will determine the best wave and its parameters based on the terrain characteristics. For example, if the robot had to pass through a tube, an amplitude smaller than the section of the tube will be needed. If the obstacle is an step, a bigger amplitude will be used.

The architecture is shown in figure 6. The controller is composed of three subsystems. *Control table* is the central part, where the angular position vectors are stored. The contents of this table determines the movement (section 3.1). The *position controller* generates the PWM signals that are applied to the servos to set their angular position.

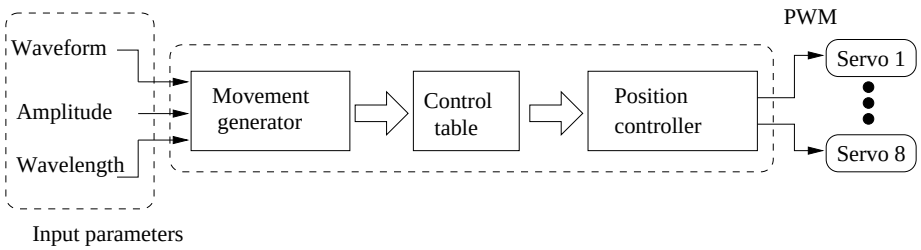


**Fig. 6.** Architecture of the locomotion controller

Finally, the *movement generator* obtain the gait control table from the parameter of the wave (waveform, amplitude and wavelength). It is implemented by software, using the algorithm described in section 3.2.

## 4 Implementation on FPGA

Mainly two different approaches can be used for the implementation of the locomotion controller:

1. Using a conventional microprocessor system, either centralized (a CPU that controls all the modules) or distributed (every module has its own embedded CPU, connected by a network). All the functionality is implemented in software. In order to add a hardware controller, a new printed circuit board design would be needed.
2. Using an FPGA system. Different hardware/software architectures can be designed and tested. Some subsystems could be implemented by hardware, while others by software.

We have focused on the second approach: a centralized FPGA systems. All the locomotion controller is embebed on the FPGA. The movement generator, as well as the control tables, are implemented by software. We have used the soft-processor Microblaze. Algorithms are coded in C language, first tested on a Linux PC and then ported to Microblaze, using the GCC Cross compiler[13], supplied by the FPGA manufacturer.

The position controller is a hardware unit, written in VHDL, that acts as a peripheral for the MicroBlaze. Software can access to this unit through ports, mapped on the main memory. The positions for the 8 servos are stored in the corresponding ports, where the position controller read them and generates the PWM signal. Then main advantage of this hardware devices is its scalability. In order to control more servos, new controllers can be mapped, without physical redesign of the board, always limited to the resources available on the FPGA: the area and pins available.

### 4.1 The Microblaze soft-processor

The MicroBlaze is soft-core 32 bit Harvard-style processor described in HDL (Hardware Description Language). It was released by Xilinx recently[10]. Figures 7 shows

the design loaded in the FPGA. The buses of the processor follow the Core Connect standard from IBM[14]. Also, a debug module has been included in order to be able to perform an intrusive debug of the processor using the GNU tool gdb[13].

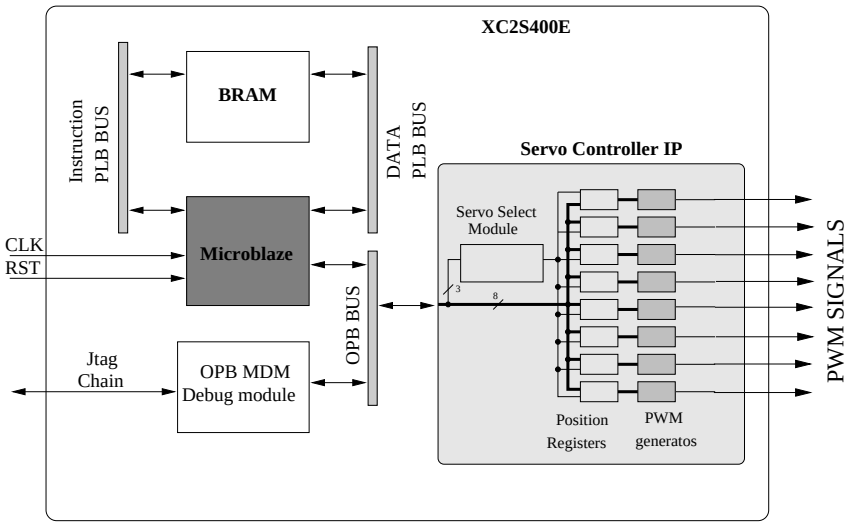


**Fig. 7.** Locomotion controller scheme loaded in the FPGA

C language can be used to design both the controller and the position computing algorithms. As the whole system (memory, buses, peripherals and processor) is being described in HDL, the hardware architecture is much simpler than traditional processor board architectures. A modification of the controller system, only needs the loading a new design into the FPGA. No PCB modification is necesary. Thus, testing and debugging stages of the design are a much simpler task. This is a fundamental feature, since robotics is a field where testing is not only a simulating task, when a modification to the hardware system is made. The Cube prototype can loads the new system in microseconds.

Traditional robotic systems have separated hardware and software design stages, the hardware system is constructed once and then the software is loaded as many times as needed to make it work. The use of an FPGA in Cube gives this robot the facility to have many hardware and software design stages so achieving desired results. It adds more flexibility in the design stages.

Finally, as the Microblaze is being designed to use very little space in the FPGA (near a 10% of a SpartanIIE400 chip is used for the Cube controlling system), all the space left can be used to implement extra hardware.

### 4.2 Implementation results

The implementation of the controlling system has been developed using the latest released Xilinx software for HDL synthesis, mapping and implementation, ISE 6.1. And the processor system developer tool, also from Xilinx EDK 6.1. The FPGA used in Cube is a SpartanIIE 400, a low cost FPGA that maintains the objective of a low cost robot. The obtained results for the final place and route of the hardware system are shown in table 1.

**Table 1.** Implementations results using an SpartanIIE 400 FPGA

| | **Total** | **Used** | **Available** |
|---|---|---|---|
| **BRAMs** | 14 | 8 | 6 (43%) |
| **Slices (Area)** | 2352 | 1312 | 1040 (44%) |
| **I/O pins** | 146 | 10 | 136 (93%) |
| **System Clock frequency (MHZ)** | – | 50 | — |

The 8 BRAM are configured to build a 32 bit words memory, having each BRAM a 4Kx4 bit capacity sharing the address bus. The results obtained for the controller leave a 44% of space and 93% of the pins free in the FPGA. So that, the system still has a remarkable amount of resources available for future improvements. The board uses a 50 MHz clock generator, even considering that no optimization of the design has been carried out.

The average robot power comsuption depends on the movement performed and will be analized in detail in future work. A Typical value is 8W (1.6A, 5v).

## 5 Conclusions and future work

A modular worm-like robot has been constructed, capable of moving in a straight line, using a wave propagation gait. Locomotion controller is based on control tables, automatically generated from the parameters of the waves applied: waveform, amplitude and wavelength. Locomotion is achieved by means of the propagation of these waves along the worm, from the tail to the head. Higher level software just need to specify this parameters to locomote the robot.

The controller has been implemented on a low cost FPGA using custom cores, described in VHDL, together with the MicroBlaze soft-processor, where the algorithms are executed. FPGAs increases the robot versatility so that the designer can select the architecture that better fix the requirements. Main limitations of this approach are the memory and FPGA resource availability. The main advandages are: possibility of implementing new architectures, faster control algorithms, dinamyc hardware modification, hardware/software codesign, and remote hardware reconfiguration.

A working platform has been developed. Current research is focused on worm locomotion, studying its characteristics as a function of the wave parameters, getting insights of its relation with velocity, stability, and consumption. One approach will be the

use of genetic algorithms to find the optimal parameters of the wave, given an stability, velocity, and power consumption restriction. We also are planning to study locomotion in a plane, not restricted only to straight lines. Finally, a new generation of modules, with embeded FPGAs are being constructed.

## Acknowledgements

This research is supported by Project Number 07T/0052/2003-3 of the *Consejería de Educación de la Comunidad Autónoma de Madrid*, Spain.

## References

1. Mark Yim, Ying Zhang & David Duff, Xerox Palo Alto Research Center (PARC), “Modular Robots”. IEEE Spectrum Magazine. Febrero 2002.
2. M. Yim, D. Duff, K.Roufas, “Modular Reconfigurable Robots, An Aproach to Urban Search and Rescue,” Proc. of 1st Intl. Workshop on Human-friendly welfare Robotic Systems (HWRS2000) Taejon, Korea, pp.69-76, Jan. 2000.
3. M. Yim, K. Roufas, D. Duff, Y. Zhang, C. Eldershaw, ”Modular Reconfigurable Robots in Space Applications”, Autonomous Robot Journal, special issue for Robots in Space, Springer Verlag, 2003.
4. D. Duff, M. Yim, K. Roufas,“Evolution of PolyBot: A Modular Reconfigurable Robot”, Proc. of the Harmonic Drive Intl. Symposium, Nagano, Japan, Nov. 2001, and Proc. of COE/Super-Mechano-Systems Workshop, Tokyo, Japan, Nov. 2001.
5. Mark Yim, David G. Duff, Kimon D. Roufas, “Polybot: a Modular Reconfigurable Robot”, IEEE intl. Conf. on Robotics and Automation (ICRA), San Francisco, CA, April 2000.
6. P. Will, A. Castano, W-M Shen, “Robot modularity for self-reconfiguration,” SPIE Intl. Symposium on Intelligent Sys. and Advanced Manufacturing, Proceeding Vol. 3839, pp.236-245, Sept. 1999.
7. K. Kotay, D. Rus, M. Vona, C. McGray, “The Self-reconfiguring Robotic-Molecule,” Proc. of the IEEE International Conf. on Robotics and Automation, pp.424-431, May 1998.
8. S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji,”A 3D self-Reconfigurable Structure,” Proc. of the IEEE International Conf. on Robotics and Automation, pp.432-439, May 1998.
9. M. Yim, Y. Zhang, K. Roufas, D. Duff, C. Eldershaw, “Connecting and disconnecting for chain self-reconfiguration with PolyBot”, IEEE/ASME Transactions on mechatronics, special issue on Information Technology in Mechatronics, 2003.
10. Xilinx inc, “Microblaze processor Reference Guide”. San Jose, California. Julio 2003.
11. Jean J. Labrosse, “Use an RTOS on your Next MicroBlaze-Based Product”. Xcell journal. Issue 48. Spring 2004.
12. Microblaze uClinux Project Home Page. [on-line] http://www.itee.uq.edu.au/~jwilliams/mblaze-uclinux/.
13. GNU project. [on-line] http://www.gnu.org.
14. IBM inc, “On-Chip Peripheral Bus, architecture specifications”. Research Triangle Park, North Carolina. April 2001.